

1. Why we need the Database? Give different database examples?

We will use the database to store the data. Different databases are Oracle, MySQL, PostgreSQL

2. What is SQL?

SQL (Structured Query Language) is a query language used to perform operations on the records stored in the database such as updating records, deleting records, creating and modifying tables, views, etc.

3. Explain the different SQL command groups?

DDL (Data Definition Language): Create, Alter and Drop

DML (Data Manipulation Language): Select, Insert, Update and Delete

DCL (Data Control Language): Grant and Revoke

TCL (transaction Control Language): Commit and Roll Back

4. Give syntax to create, rename, drop and select a database?

Create: CREATE DATABASE database_name;

Example: create database student;

Rename: ALTER DATABASE old_name MODIFY NAME = new_name

Example: alter database 'student' modify name='student_teacher';

Drop: DROP DATABASE database_name;

Example: drop database student_teacher;

Select: USE DATABASE database_name;

Example: use database student;

Note: If you delete or drop the database, all the tables and views will also be deleted. So be careful while using this command

5. Give syntax to create, delete, drop, rename, copy, truncate and alter the table?

Create Table Syntax:

```
SQL> CREATE TABLE STUDENTS (  
  ID INT NOT NULL,  
  NAME VARCHAR (20) NOT NULL,  
  AGE INT NOT NULL,  
  ADDRESS CHAR (25),  
  PRIMARY KEY (ID)  
);
```

Drop Table Syntax:

```
SQL> DROP TABLE STUDENTS;
```

Note: DROP TABLE statement is used to delete a table definition and all data from a table. Once a table is deleted all the information available in the table is lost forever, so we have to be very careful when using this command.

Delete table Syntax:

```
DELETE FROM STUDENTS WHERE student_name='Alfreds Futterkiste';
```

Note: The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

Alter Table Syntax:

The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table. You should also use the ALTER TABLE command to add and drop various constraints on an existing table.

Add new Column:

```
ALTER TABLE table_name ADD column_name datatype;
```

Modify Column:

```
ALTER TABLE table_name MODIFY column_name column_type;
```

Drop Column:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Add Primary key to column:

```
ALTER TABLE table_name  
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

Delete Primary key on column:

```
ALTER TABLE table_name  
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

Truncate Table Syntax:

```
TRUNCATE TABLE Employee;
```

A truncate SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause. The rollback process is not possible after truncate table statement. Once you truncate a table you cannot use a flashback table statement to retrieve the content of the table.

Copy Table Syntax:

```
SELECT * INTO admin_employee FROM hr_employee;
```

If you want to copy a SQL table into another table in the same SQL server database.

Rename Table Name Syntax:

```
ALTER TABLE table_name  
RENAME TO new_table_name;
```

6. Explain about Joins?

JOIN means "to combine two or more tables". Below are the types of joins.

INNER JOIN – returns rows when there is a match in both tables.

LEFT JOIN – returns all rows from the left table, even if there are no matches in the right table.

RIGHT JOIN – returns all rows from the right table, even if there are no matches in the left table.

FULL JOIN – SQL full outer join is used to combine the result of both left and right outer join and returns all rows (don't care its matched or unmatched) from the both participating tables.

CROSS JOIN – returns the Cartesian product of the sets of records from the two or more joined tables.

INNER Join Example:

```
SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

It will returns rows when there is a match in both tables.

LEFT Join Example:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

It will return all records from left table and matching records from right table. If no records matching in right table it won't care of it.

RIGHT Join Example:

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

It will return all records from right table and matching records from left table. If no records matching in left table it won't care of it.

Full Join Example:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Self Join:

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

A self join is a regular join, but the table is joined with itself.

7. What is the use of inner/sub queries?

A Sub query or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A sub query is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

In independent nested queries, query execution starts from innermost query to outermost queries. The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query.

Sub queries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, let us check the following subquery with a SELECT statement.

```
SQL> SELECT *  
      FROM CUSTOMERS  
      WHERE ID IN (SELECT ID  
                  FROM CUSTOMERS  
                  WHERE SALARY > 4500) ;
```

This would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

8. Explain about the SQL Constraints?

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a **NOT NULL** and **UNIQUE**. Uniquely identifies each row in a table
- **FOREIGN KEY** - Prevents actions that would destroy links between tables
- **CHECK** - Ensures that the values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column if no value is specified
- **CREATE INDEX** - Used to create and retrieve data from the database very quickly

9. Give an example to add constraints on table?

To add the constraints on columns or tables we can use the alter command on existing tables or we can add constraints during table creation time also.

Foreign Key:

```
CREATE TABLE orders
(
  O_Id int NOT NULL,
  Order_No int NOT NULL,
  S_Id int,
  PRIMARY KEY (O_Id),
  FOREIGN KEY (S_Id) REFERENCES Persons (S_Id)
)
```

10. Explain about composite primary key?

A composite key is a combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness is guaranteed, but when it taken individually it does not guarantee uniqueness.

```
CREATE TABLE SAMPLE_TABLE
(COL1 integer,
COL2 varchar(30),
COL3 varchar(50),
PRIMARY KEY (COL1, COL2));
```

11. What is the use of unions?

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

```
SELECT column1 [, column2 ]  
FROM table1 [, table2 ]  
[WHERE condition]  
  
UNION  
  
SELECT column1 [, column2 ]  
FROM table1 [, table2 ]  
[WHERE condition]
```

Union: The UNION operator is used to combine the results of two SELECT statements without including duplicate rows.

Union All: The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows.

12. What is the use of views?

- ✓ In SQL, View can be described as virtual table which derived its data from one or more than one table columns.
- ✓ A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- ✓ You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

To create an view

```
CREATE OR REPLACE VIEW view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

To drop an view,

```
DROP VIEW view_name;
```

13. Advantages of Views?

Security: Each user can be given permission to access the database only through a small set of views that contain the specific data the user is authorized to see, thus restricting the user's access to stored data.

Query Simplicity: A view can draw data from several different tables and present it as a single table, turning multi-table queries into single-table queries against the view.

14. What is meant by temporary tables?

There are RDBMS, which support temporary tables. Temporary Tables are a great feature that lets you store and process intermediate results by using the same selection, update, and join capabilities that you can use with typical SQL Server tables.

The temporary tables could be very useful in some cases to keep temporary data. The most important thing that should be known for temporary tables is that they will be deleted when the current client session terminates.

```
CREATE TEMPORARY TABLE table_name(  
    column_1_definition,  
    column_2_definition,  
    ...,  
    table_constraints  
);
```

15. Difference between View and Temporary table?

The main difference between temporary tables and views is that temporary tables are just the tables in tempdb, but views are just stored queries for existing data in existing tables. So, there is no need to populate the view, because the data is already here. But temporary table needs to be populated first, and population is the main performance-concerned issue.

16. Explain about indexes?

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

The **CREATE INDEX** statement is used to create indexes in tables.

```
CREATE INDEX index_name  
ON table_name (column1, column2, ...);
```

Drop index:

```
DROP INDEX index_name ON table_name;
```


Index types:

Single-Column Indexes

A single-column index is created based on only one table column. The basic syntax is as follows.

```
CREATE INDEX index_name  
ON table_name (column_name);
```

Unique Indexes

Unique indexes are used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table. The basic syntax is as follows.

```
CREATE UNIQUE INDEX index_name  
on table_name (column_name);
```

Composite Indexes

A composite index is an index on two or more columns of a table. Its basic syntax is as follows.

```
CREATE INDEX index_name  
on table_name (column1, column2);
```

Note:

An index helps to speed up **SELECT** queries and **WHERE** clauses, but it slows down data input, with the **UPDATE** and the **INSERT** statements. Indexes can be created or dropped with no effect on the data.

17. Give an example for Insert Query?

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)  
VALUES (value1, value2, value3,...valueN);
```

Way to populate the one table data into other by using insert query,

```
INSERT INTO first_table_name [(column1, column2, ... columnN)]  
SELECT column1, column2, ...columnN  
FROM second_table_name  
[WHERE condition];
```

18. Given an example for update Query?

```
UPDATE table_name  
SET column1 = value1, column2 = value2..., columnN = valueN  
WHERE [condition];
```

Example:

```
SQL> UPDATE CUSTOMERS  
SET ADDRESS = 'Pune'  
WHERE ID = 6;
```

Note: If you want to update the same data for all columns takeout where condition.

19. Give an example for delete query?

```
DELETE FROM table_name  
WHERE [condition];
```

Example:

```
SQL> DELETE FROM CUSTOMERS  
WHERE ID = 6;
```

Note: If you want to delete the all data from table takeout where condition.

20. Difference between Drop, Delete and truncate on table?

DROP is a DDL Command. Objects deleted using DROP are permanently lost and it cannot be rolled back.

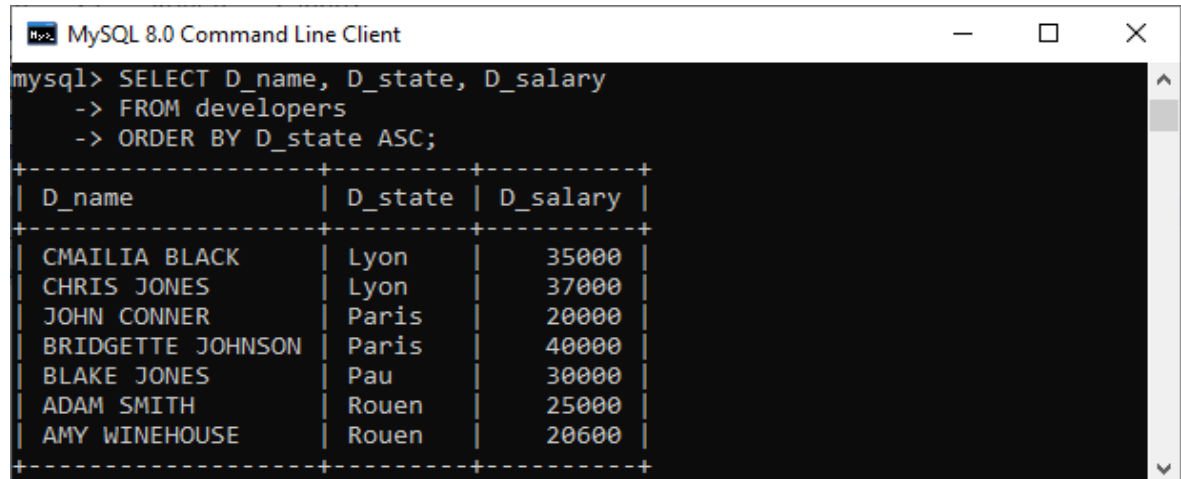
TRUNCATE Command is a Data Definition Language operation. It is used to remove all the records from a table. It deletes all the records from an existing table but not the table itself. The structure or schema of the table is preserved.

DELETE statement in SQL is a Data Manipulation Language (DML) Command. It is used to delete existing records from an existing table. We can delete a single record or multiple records depending on the condition specified in the query

The DELETE statement scans every row before deleting it. Thus it is slower as compared to TRUNCATE command. If we want to delete all the records of a table, it is preferable to use TRUNCATE in place of DELETE as the former is faster than the latter.

21. Difference between group by and order by?

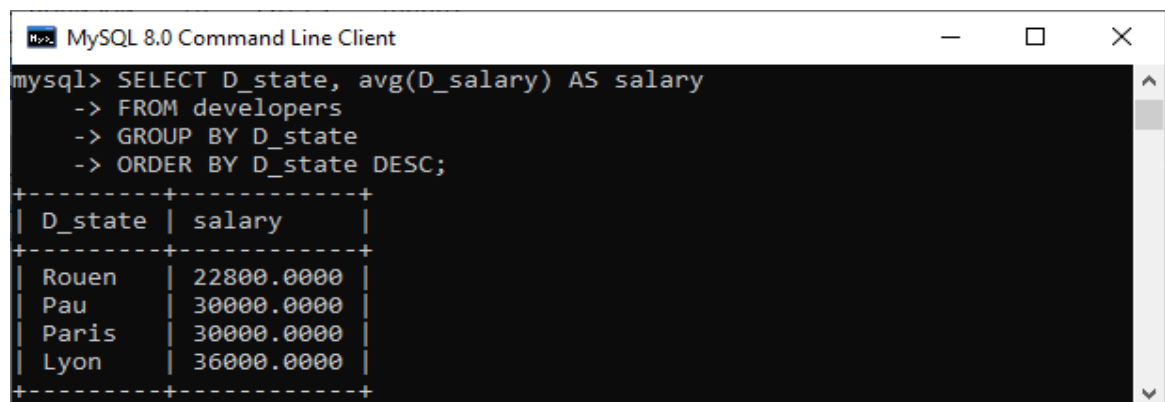
GROUP BY clause is applicable when we want to use aggregate functions to more than one set of rows.



```
mysql> SELECT D_name, D_state, D_salary
-> FROM developers
-> ORDER BY D_state ASC;
```

D_name	D_state	D_salary
CMAILIA BLACK	Lyon	35000
CHRIS JONES	Lyon	37000
JOHN CONNER	Paris	20000
BRIDGETTE JOHNSON	Paris	40000
BLAKE JONES	Pau	30000
ADAM SMITH	Rouen	25000
AMY WINEHOUSE	Rouen	20600

ORDER BY clause is applicable when we want to get the data obtained by a query in the sorting order.



```
mysql> SELECT D_state, avg(D_salary) AS salary
-> FROM developers
-> GROUP BY D_state
-> ORDER BY D_state DESC;
```

D_state	salary
Rouen	22800.0000
Pau	30000.0000
Paris	30000.0000
Lyon	36000.0000

22. Explain about stored procedures?

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

Header: The header contains the name of the procedure and the parameters or variables passed to the procedure.

Body: The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

We can pass the parameters like below to the procedure,

IN

An IN parameter lets you pass a value to the subprogram. **It is a read-only parameter.** Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a constant, literal, initialized variable, or expression as an IN parameter. You can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. **It is the default mode of parameter passing. Parameters are passed by reference.**

OUT

An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. **The actual parameter must be variable and it is passed by value.**

IN OUT

An **IN OUT** parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read.

The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. **Actual parameter is passed by value.**

Syntax:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
    [ (parameter [,parameter]) ]
IS
    [declaration_section]
BEGIN
    executable_section
[EXCEPTION
    exception_section]
END [procedure_name];
```

Create procedure example

In this example, we are going to insert record in user table. So you need to create user table first.

Table creation:

```
create table user(id number(10) primary key,name varchar2(100));
```

Now write the procedure code to insert record in user table.

Procedure Code:

```
create or replace procedure "INSERTUSER"  
(id IN NUMBER,  
 name IN VARCHAR2)  
is  
begin  
insert into user values(id,name);  
end;  
/
```

A procedure may or may not return any value.

23. Explain about Functions?

The PL/SQL Function is very similar to PL/SQL Procedure. The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value. Except this, all the other things of PL/SQL procedure are true for PL/SQL function too.

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]  
[(parameter_name [IN | OUT | IN OUT] type [, ...])]   
RETURN return_datatype  
{IS | AS}  
BEGIN  
    < function_body >  
END [function_name];
```

Example:

```

create or replace function adder(n1 in number, n2 in number)
return number
is
n3 number(8);
begin
n3 :=n1+n2;
return n3;
end;
/

```

24. Explain about Triggers?

Trigger is invoked by Oracle engine automatically whenever a specified event occurs. Trigger is stored into database and invoked repeatedly, when specific condition match.

Triggers are stored programs, which are automatically executed or fired when some event occurs.

Triggers are written to be executed in response to any of the following events.

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

Syntax:

```

CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
    Declaration-statements
BEGIN
    Executable-statements
EXCEPTION
    Exception-handling-statements
END;

```

Example:

```
Select * from customers;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.salary - :OLD.salary;
    dbms_output.put_line('Old salary: ' || :OLD.salary);
    dbms_output.put_line('New salary: ' || :NEW.salary);
    dbms_output.put_line('Salary difference: ' || sal_diff);
END;
```

Triggering a Trigger

Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table –

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (7, 'Kriti', 22, 'HP', 7500.00 );
```

When a record is created in the CUSTOMERS table, the above create trigger, **display_salary_changes** will be fired and it will display the following result –

```
Old salary:
New salary: 7500
Salary difference:
```

Because this is a new record, old salary is not available and the above result comes as null. Let us now perform one more DML operation on the CUSTOMERS table. The UPDATE statement will update an existing record in the table –

```
UPDATE customers
SET salary = salary + 500
WHERE id = 2;
```

When a record is updated in the CUSTOMERS table, the above create trigger, **display_salary_changes** will be fired and it will display the following result –

```
Old salary: 1500
New salary: 2000
Salary difference: 500
```

25. Explain about packages?

PL/SQL package is a logical grouping of a related subprogram (procedure/function) into a single element. A Package is compiled and stored as a database object that can be used later.

Packages have two components,

- ✓ **Package Specification:** consists of a declaration of all the public variables, cursors, objects, procedures, functions, and exception.
- ✓ **Package Body:** It consists of the definition of all the elements that are present in the package specification. It can also have a definition of elements that are not declared in the specification, these elements are called private elements and can be called only from inside the package.

26. Explain about cursors?

When an SQL statement is processed, Oracle creates a memory area known as context area. A cursor is a pointer to this context area. It contains all information needed for processing the statement. In PL/SQL, the context area is controlled by Cursor. A cursor contains information on a select statement and the rows of data accessed by it.

Types of cursors:

Implicit: The implicit cursors are automatically generated by Oracle while an SQL statement is executed, if you don't use an explicit cursor for the statement. Oracle provides some attributes known as implicit cursor's attributes to check the status of DML operations. Some of them are: **%FOUND, %NOTFOUND, %ROWCOUNT and %ISOPEN**. Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

```
DECLARE
    total_rows number(2);
BEGIN
    UPDATE customers
    SET salary = salary + 500;
    IF sql%notfound THEN
        dbms_output.put_line('no customers selected');
    ELSIF sql%found THEN
        total_rows := sql%rowcount;
        dbms_output.put_line( total_rows || ' customers selected ');
    END IF;
END;
/
```


Explicit: Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

Syntax:

```
CURSOR cursor_name IS select_statement;
```

Example:

```
CURSOR c_customers IS  
  SELECT id, name, address FROM customers;
```

27. Explain about exceptions?

An error occurs during the program execution is called Exception in PL/SQL.

PL/SQL facilitates programmers to catch such conditions using exception block in the program and an appropriate action is taken against the error condition.

Types of exceptions:

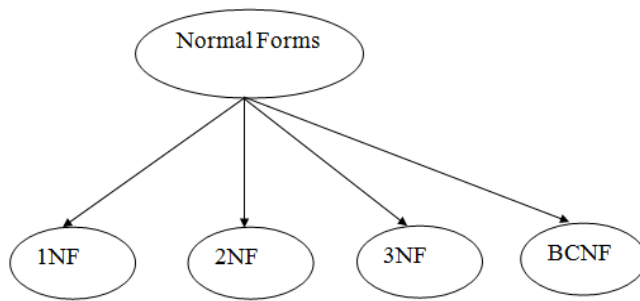
- ✓ System-defined Exceptions
- ✓ User-defined Exceptions

```
DECLARE  
  <declarations section>  
BEGIN  
  <executable command(s)>  
EXCEPTION  
  <exception handling goes here >  
  WHEN exception1 THEN  
    exception1-handling-statements  
  WHEN exception2 THEN  
    exception2-handling-statements  
  WHEN exception3 THEN  
    exception3-handling-statements  
  .....  
  WHEN others THEN  
    exception3-handling-statements  
END;
```

28. Explain about Normalization?

Normalization is the process of organizing the data in the database. Normalization divides the larger table into the smaller table and links them using relationship. The normal form is used to reduce redundancy from the database table.

Types of Normalization:



1NF: A relation will be 1NF if it contains an atomic value. It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

2NF: In the 2NF, relational must be in 1NF. In the second normal form, all non-key attributes are fully functional dependent on the primary key.

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

After 2NF applied,

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

3 NF: A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency. 3NF is used to reduce the data duplication.

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

After applying the 3 NF:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Boyce Codd normal form: It is extension of third normal form. And, for any dependency $A \rightarrow B$, A should be a super key

student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

After applying Boyce Codd NF:

Student Table

student_id	p_id
101	1
101	2
and so on...	

And, Professor Table

p_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++

29. Write an SQL query to get the third maximum salary of an employee from a table named employee_table?

```
SELECT TOP 1 salary
FROM (
SELECT TOP 3 salary
FROM employee_table
ORDER BY salary DESC ) AS emp
ORDER BY salary ASC;
```

30. Explain about having clause?

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

```
SQL > SELECT ID, NAME, AGE, ADDRESS, SALARY  
FROM CUSTOMERS  
GROUP BY age  
HAVING COUNT(age) >= 2;
```

	ID		NAME		AGE
	2		Khilan		25
			Delhi		1500.00